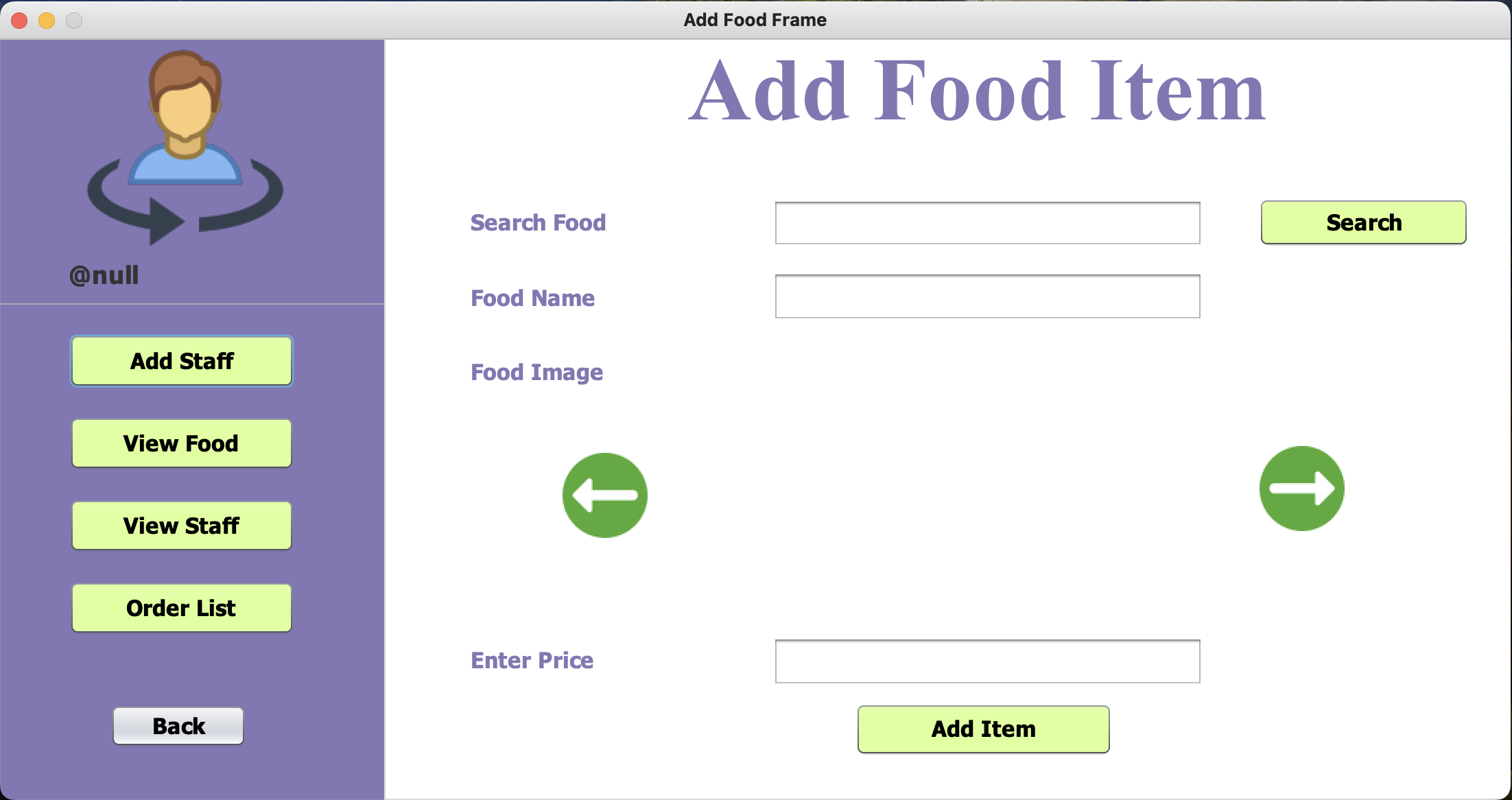
**Java SE Project**

YumXpress Application

***Add Food Frame Documentation***

1. **AddFoodFrame -**

**2. STEPS TO BE DONE IN** **AddFoodFrame**

In the **AddFoodFrame** we need to do following steps:

**A. Creating the Products Table**

**B. Creating ProductPojo**

**C. Creating ProductDao with 2 methods: getNewProductId() and addProduct()**

**D. Creating FoodApi interface with 2 methods: getFoodItemList() and getFoodItemById()**

**E. Creating SpoonacularFoodApi class by implementing the FoodApi interface**

**F. Event handler for Search button**

**G. Event handler for next and preview icon**

**H. Coding for Add Item Button:**

**I. Handling all the options :**

**Command :**

**CREATE TABLE PRODUCTS (**

**PRODUCT\_ID VARCHAR2(10) CONSTRAINT PRODUCTS\_PK PRIMARY KEY,**

**COMPANY\_ID VARCHAR2(10) CONSTRAINT PD\_FK\_CMP\_ID FOREIGN KEY REFERENCES COMPANIES(COMPANY\_ID),**

**PRODUCT\_NAME VARCHAR2(50),**

**PRODUCT\_PRICE NUMBER(7,2),**

**PRODUCT\_IMAGE BLOB**

**);**

**Coding of ProductPojo class:**

**public class ProductPojo {**

**private String productId;**

**private String companyId;**

**private String productName;**

**private double productPrice;**

**private Image productImage;**

**private String productImageType;**

**public ProductPojo() {**

**}**

**public ProductPojo(String productId, String companyId, String productName, double productPrice, Image productImage, String productImageType) {**

**this.productId = productId;**

**this.companyId = companyId;**

**this.productName = productName;**

**this.productPrice = productPrice;**

**this.productImage = productImage;**

**this.productImageType = productImageType;**

**}**

**public String getProductId() {**

**return productId;**

**}**

**public void setProductId(String productId) {**

**this.productId = productId;**

**}**

**public String getCompanyId() {**

**return companyId;**

**}**

**public void setCompanyId(String companyId) {**

**this.companyId = companyId;**

**}**

**public String getProductName() {**

**return productName;**

**}**

**public void setProductName(String productName) {**

**this.productName = productName;**

**}**

**public double getProductPrice() {**

**return productPrice;**

**}**

**public void setProductPrice(double productPrice) {**

**this.productPrice = productPrice;**

**}**

**public Image getProductImage() {**

**return productImage;**

**}**

**public void setProductImage(Image productImage) {**

**this.productImage = productImage;**

**}**

**public String getProductImageType() {**

**return productImageType;**

**}**

**public void setProductImageType(String productImageType) {**

**this.productImageType = productImageType;**

**}**

**@Override**

**public String toString() {**

**return "ProductPojo{" + "productId=" + productId + ", companyId=" + companyId + ", productName=" + productName + ", productPrice=" + productPrice + ", productImage=" + productImage + ", productImageType=" + productImageType + '}';**

**}**

**}**

**Coding of addProduct() method:**

**public static boolean addProduct(ProductPojo product) throws SQLException, IOException {**

**// Create a BufferedImage object**

**BufferedImage bufferedImage**

**= new BufferedImage(product.getProductImage().getWidth(null), product.getProductImage().getHeight(null),**

**BufferedImage.TYPE\_INT\_RGB);**

**// Draw the image inside BufferedImage**

**Graphics gr= bufferedImage.getGraphics();**

**gr.drawImage(product.getProductImage(), 0, 0, null);**

**// Convert BufferedImage to byte array**

**ByteArrayOutputStream baos = new ByteArrayOutputStream();**

**ImageIO.write(bufferedImage, product.getProductImageType(), baos);**

**byte[] imageData = baos.toByteArray();**

**// sending the data to the database;**

**Connection conn = DBConnection.getConnection();**

**PreparedStatement ps = conn.prepareStatement("Insert into products values(?,?,?,?,?)");**

**ps.setString(1, getNewProductId());**

**ps.setString(2, product.getCompanyId());**

**ps.setString(3, product.getProductName());**

**ps.setDouble(4, product.getProductPrice());**

**ps.setBinaryStream(5, new ByteArrayInputStream(imageData), imageData.length);**

**int x = ps.executeUpdate();**

**return x > 0;**

**}**

Let’s discuss about **Spoonacular** api  
===========================  
1. The **spoonacular api** allows us to access over **800,000 food products**, **380,000 recipes** and thousands of ingredients.

**To use this api we first have to signup by using the below link for the api key:**  
[https://spoonacular.com/food-api/console#](https://spoonacular.com/food-api/console)  
  
A. After registering, do the following steps:

1. Go to MY CONSOLE menu  
 2. Click on Profile  
 3. Here you can see API Key

B. Now, we have to fire an api endPoint:  
 **String endPoint = "https://api.spoonacular.com/food/search?query=" + searchKey + "&number=2&apiKey=" + apiKey;**

C. By replacing **searchKey** with food name and **apiKey** with apiKey.

D. After that we will get response in **JSON format.**

**This will be an array of JSON objects so we need to do the following:**

1. **Iterate over this array and get one object at a time from it.**
2. **This object will contain a member called results which itself is an array holding food name and it’s image url along with other details.**
3. **We extract the food name and image url from it.**
4. **Then we again send a request to the SERVER for getting the actual image represented by the image url.**
5. **We then convert this image into a java.awt.Image object.**
6. **Then we create a ProductPojo object and store Food Name, It’s Image and Image type in that Pojo.**
7. **We add this Pojo to an ArrayList**
8. **After all the food names have been accessed and added in the ArrayList , we finally return this ArrayList back to the front end**

**E. To do the above steps (B, C, and D) in Core Java (JSE), we have to download some libraries:** 1. unirest-java-1.4.7.jar

2. json-20140107.jar

**Practicing API Call**

**===============**

**public class GetApiData {**

**public static void main(String[] args) {**

**String apiKey = "9763f81b0cb4487285d9cd5624c14bdd";**

**Scanner kb=new Scanner(System.in);**

**System.out.println("Enter food name");**

**String searchKey=kb.nextLine();**

**String apiUrl = " https://api.spoonacular.com/food/search?query=" + searchKey + "&number=2&apiKey=" + apiKey;**

**try {**

**HttpResponse<JsonNode> response = Unirest.get(apiUrl).header("accept", "application/json").asJson();**

**JSONObject jsonResponse = response.getBody().getObject();**

**JSONArray searchResultsArray = jsonResponse.getJSONArray("searchResults");**

**//System.out.println("SearchResultsArray:"+searchResultsArray);**

**for (int i = 0; i < searchResultsArray.length(); i++) {**

**JSONObject searchResult = searchResultsArray.getJSONObject(i);**

**JSONArray resultsArray = searchResult.getJSONArray("results");**

**for (int j = 0; j < resultsArray.length(); j++) {**

**JSONObject result = resultsArray.getJSONObject(j);**

**if (!result.has("name") || !result.has("image")) {**

**continue;**

**}**

**String foodName = result.getString("name");**

**String imageUrl = result.getString("image");**

**String imageType = imageUrl.substring(imageUrl.lastIndexOf(".")+1, imageUrl.length());**

**if (foodName.isEmpty() || imageUrl.isEmpty()) {**

**continue;**

**}**

**HttpURLConnection url = (HttpURLConnection) new URL(imageUrl).openConnection();**

**url.addRequestProperty("user-agent", "mozilla");**

**if (url.getResponseCode() != HttpURLConnection.HTTP\_OK) {**

**continue;**

**}**

**Image image = ImageIO.read(url.getInputStream());**

**System.out.println("food name:"+foodName + " - image url " + imageUrl);**

**System.out.println("image type is "+imageType);**

**}**

**}**

**}catch (UnirestException | IOException ex) {**

**ex.printStackTrace();**

**}**

**}**

**}**

**Coding for FoodApi interface in the utility package:**

public interface FoodApi {

public ArrayList<ProductPojo> getFoodItemList(String searchKey);

public ImageIcon getFoodItemById(int foodId);

}

**Coding for SpoonacularFoodApi class in the utility package:**

public class SpoonacularFoodApi implements FoodApi {

private ArrayList<ProductPojo> productList = new ArrayList<>();

private ProductPojo product;

@Override

public ArrayList<ProductPojo> getFoodItemList(String searchKey) {

String apiKey = “9763f81b0cb448\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*”;

String apiUrl = " https://api.spoonacular.com/food/search?query=" + **searchKey** + "&number=2&apiKey=" + **apiKey**;

try {

HttpResponse<JsonNode> response = Unirest.get(apiUrl)

.header("accept", "application/json")

.asJson();

JSONObject jsonResponse = response.getBody().getObject();

JSONArray searchResultsArray = jsonResponse.getJSONArray("searchResults");

for (int i = 0; i < searchResultsArray.length(); i++) {

JSONObject searchResult = searchResultsArray.getJSONObject(i);

JSONArray resultsArray = searchResult.getJSONArray("results");

**// Do something with the resultsArray, such as display the name and image for each result**

for (int j = 0; j < resultsArray.length(); j++) {

JSONObject result = resultsArray.getJSONObject(j);

if (!result.has("name") || !result.has("image")) {

continue;

}

String foodName = result.getString("name");

String imageUrl = result.getString("image");

String imageType = imageUrl.substring(imageUrl.lastIndexOf(".")+1, imageUrl.length());

if (foodName.isEmpty() || imageUrl.isEmpty()) {

continue;

}

HttpURLConnection url = (HttpURLConnection) new URL(imageUrl).openConnection();

url.addRequestProperty("user-agent", "mozilla");

if (url.getResponseCode() != HttpURLConnection.HTTP\_OK) {

continue;

}

Image image = ImageIO.read(url.getInputStream());

System.out.println("i can print befroe expection" + image);

System.out.println(foodName + " - " + imageUrl);

System.out.println("image type is "+imageType);

// add HttpURLConnection url and product name in the hashmap of arraylist

product = new ProductPojo();

product.setProductName(foodName);

product.setProductImage(image);

product.setProductImageType(imageType);

productList.add(product);

}

}

} catch (UnirestException | IOException ex) {

ex.printStackTrace();

}

return productList;

}

@Override

public ImageIcon getFoodItemById(int foodId) {

String foodIdUrl = "https://api.spoonacular.com/food/menuItems/" + foodId + "?apiKey=9763f81b0cb4487285d9cd5624c14bdd";

HttpResponse<JsonNode> response;

ImageIcon imageIcon = null;

try {

response = Unirest.get(foodIdUrl)

.header("accept", "application/json")

.asJson();

JSONObject jsonResponse = response.getBody().getObject();

System.out.println("JSON Response : "+jsonResponse);

String foodName = jsonResponse.getString("title");

String imageUrl = jsonResponse.getString("image");

System.out.println("Food ID: " + foodId);

System.out.println("Food Name: " + foodName);

System.out.println("Image URL: " + imageUrl);

HttpURLConnection url = (HttpURLConnection) new URL(imageUrl).openConnection();

url.addRequestProperty("user-agent", "mozilla");

Image image = ImageIO.read(url.getInputStream());

System.out.println("i can print befroe expection" + image);

imageIcon = new ImageIcon(image);

} catch (UnirestException ex) {

Logger.getLogger(SpoonacularFoodApi.class.getName()).log(Level.SEVERE, null, ex);

} catch (MalformedURLException ex) {

Logger.getLogger(SpoonacularFoodApi.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

Logger.getLogger(SpoonacularFoodApi.class.getName()).log(Level.SEVERE, null, ex);

}

return imageIcon;

}

}

**F. Coding for Search Button :**

**private void btnSearchFoodActionPerformed(java.awt.event.ActionEvent evt) {**

**if (txtSearchFood.getText().trim().isEmpty()) {**

**JOptionPane.showMessageDialog(this, "Please Give The Food Name To Be Searched.", "Error", JOptionPane.ERROR\_MESSAGE);**

**return;**

**}**

**String searchFood = txtSearchFood.getText().trim().replace(' ', '-');**

**System.out.println("food to be search " + searchFood);**

**if (productList != null) {**

**productList.clear();**

**}**

**try {**

**productList = spoonacularApi.getFoodItemList(searchFood);**

**} catch (Exception ex) {**

**JOptionPane.showMessageDialog(this, "Couldn't Load This Product.", "Error", JOptionPane.ERROR\_MESSAGE);**

**return;**

**}**

**lblNext.setEnabled(true);**

**lblBack.setEnabled(true);**

**showFoodDetailsOfIndex(viewImageIndex);**

**}**

**Coding for showFoodDetailsOfIndex() method :**

private void showFoodDetailsOfIndex(int viewImageIndex) {

product = productList.get(viewImageIndex);

String productName = product.getProductName();

if (productName.length() >= 50) {

productName = productName.substring(0, 49);

}

txtFoodName.setText(productName);

ImageIcon imageIcon = new ImageIcon(product.getProductImage()

.getScaledInstance(lblFoodImage.getWidth(), lblFoodImage.getHeight(), Image.SCALE\_SMOOTH));

lblFoodImage.setIcon(imageIcon);

}

By getting response from the spoonacular api we will set food title in text box of food name, set food image on the label, and set the price of food. Also we can change food name, food image using icon arrow and price of the food.

**G. Coding for next and preview image icon**

**previous icon:**

private void lblBackMouseClicked(java.awt.event.MouseEvent evt) {

viewImageIndex--;

if (viewImageIndex < 0) {

viewImageIndex = productList.size() - 1;

}

System.out.println("mouse clicked left ..... ====" + viewImageIndex);

showFoodDetailsOfIndex(viewImageIndex);

}

**next icon:**

private void lblNextMouseClicked(java.awt.event.MouseEvent evt) {

viewImageIndex++;

if (viewImageIndex >= productList.size()) {

viewImageIndex = 0;

}

System.out.println("mouse clicked right ......... ====" + viewImageIndex);

showFoodDetailsOfIndex(viewImageIndex);

}

**H. Coding for Add Item Button:**

On the click of Add Item button we will validate input details and store all the data in the ProductPojo class and add the product in the PRODUCTS table by calling the static method addProduct() of ProductDao class.

**private void btnAddItemActionPerformed(java.awt.event.ActionEvent evt) {**

**if (validateInput()) {**

**JOptionPane.showMessageDialog(this, "Please Fill All The Fields.", "Error", JOptionPane.ERROR\_MESSAGE);**

**return;**

**}**

**try {**

**price = Double.parseDouble(txtFoodPrice.getText().trim());**

**} catch (NumberFormatException nfe) {**

**nfe.printStackTrace();**

**JOptionPane.showMessageDialog(this, "Please Enter Numbers only.", "Error", JOptionPane.ERROR\_MESSAGE);**

**return;**

**}**

**try {**

**product.setCompanyId(OwnerProfile.getCompanyId());**

**product.setProductName(txtFoodName.getText().toUpperCase());**

**product.setProductPrice(price);**

**boolean isProductAdded = ProductDao.addProduct(product);**

**System.out.println("product added : " + isProductAdded);**

**if (!isProductAdded) {**

**JOptionPane.showMessageDialog(this, "Can't Add The Product Now.\nTry Again Later", "Error", JOptionPane.ERROR\_MESSAGE);**

**return;**

**}**

**JOptionPane.showMessageDialog(this, "Product Added Successfully", "Success", JOptionPane.INFORMATION\_MESSAGE);**

**productList.remove(viewImageIndex);**

**if(productList.isEmpty()){**

**clearAll();**

**JOptionPane.showMessageDialog(this, "All food types for"+txtFoodName.getText()+" added.");**

**return;**

**}**

**viewImageIndex = 0;**

**showFoodDetailsOfIndex(viewImageIndex);**

**txtFoodPrice.setText("");**

**} catch (SQLException | IOException ex) {**

**ex.printStackTrace();**

**}**

**}**

**Coding of clearAll()**

**private void clearAll() {**

**txtSearchFood.setText("");**

**txtFoodName.setText("");**

**txtFoodPrice.setText("");**

**lblFoodImage.setIcon(null);**

**lblNext.setEnabled(false);**

**lblBack.setEnabled(false);**

**}**

**Coding of validateInput() method:**

**private boolean validateInput() {**

**if (txtFoodName.getText().trim().equals("") || txtFoodPrice.getText().trim().equals("") || lblFoodImage.getIcon() == null) {**

**return true;**

**}**

**return false;**

**}**

**I. Handling all the options :**

We will create the ActionListener Events for every option. In every clicked button will be opening the respected frame and close this frame.

**Coding for handling all the options :**

**private void btnViewStaffActionPerformed(java.awt.event.ActionEvent evt) {**

**showFrame = new ViewDeliveryStaffFrame();**

**showFrame.setVisible(Boolean.TRUE);**

**this.dispose();**

**}**

**private void btnViewFoodActionPerformed(java.awt.event.ActionEvent evt) {**

**showFrame = new ViewFoodFrame();**

**showFrame.setVisible(Boolean.TRUE);**

**this.dispose();**

**}**

**private void btnViewStaffActionPerformed(java.awt.event.ActionEvent evt) {**

**showFrame = new ViewDeliveryStaffFrame();**

**showFrame.setVisible(Boolean.TRUE);**

**this.dispose();**

**}**

**private void btnOrderListActionPerformed(java.awt.event.ActionEvent evt) {**

**// TODO add your handling code here:**

**showFrame = new OrderListFrame();**

**showFrame.setVisible(Boolean.TRUE);**

**this.dispose();**

**}**

**private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {**

**showFrame=new SellerOptionFrame();**

**showFrame.setVisible(true);**

**this.dispose();**

**}**